PATENT & TRADEMARK OFFICE
FEB 19 2003
OIPE
JC100

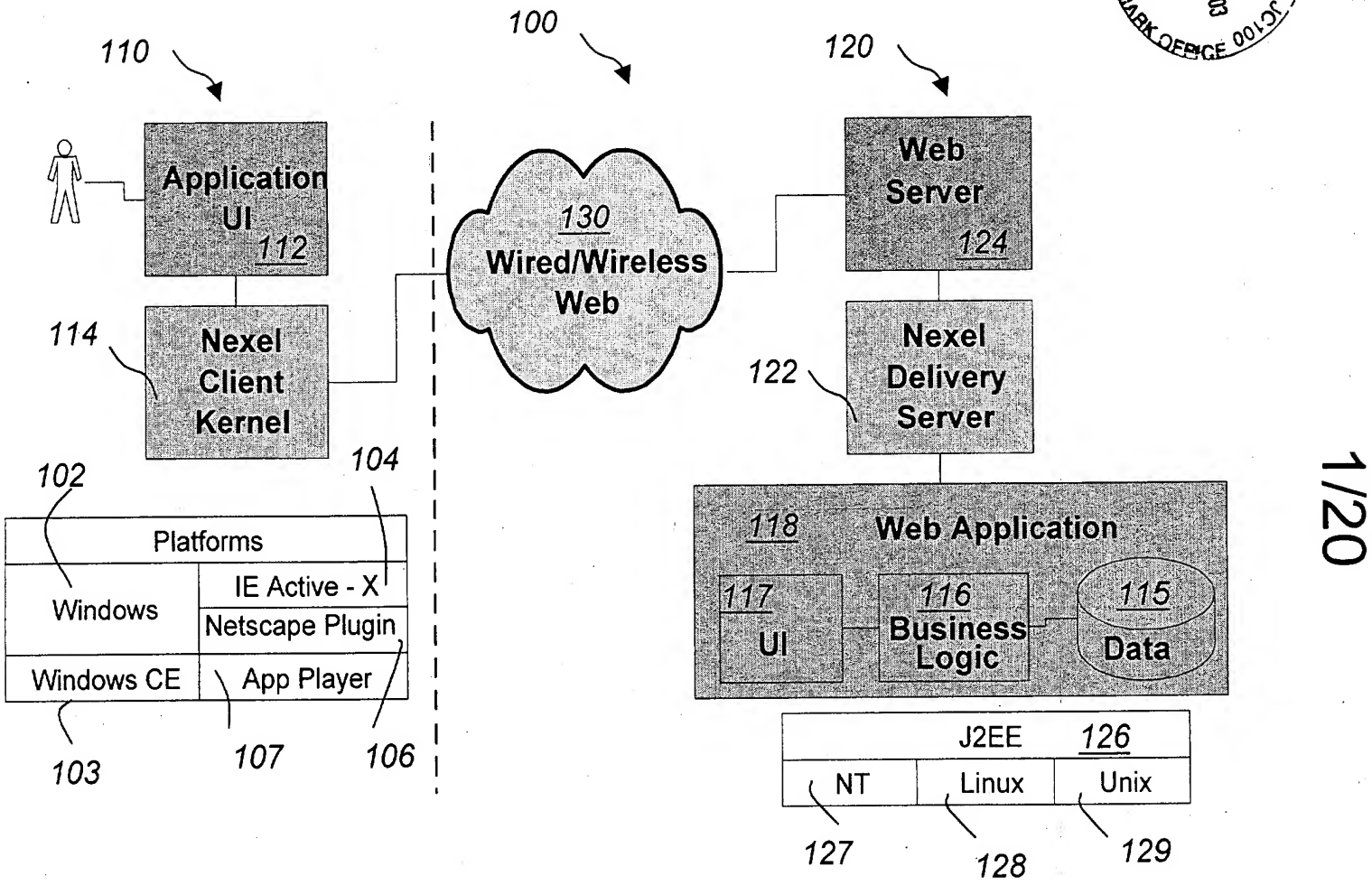


FIG. 1

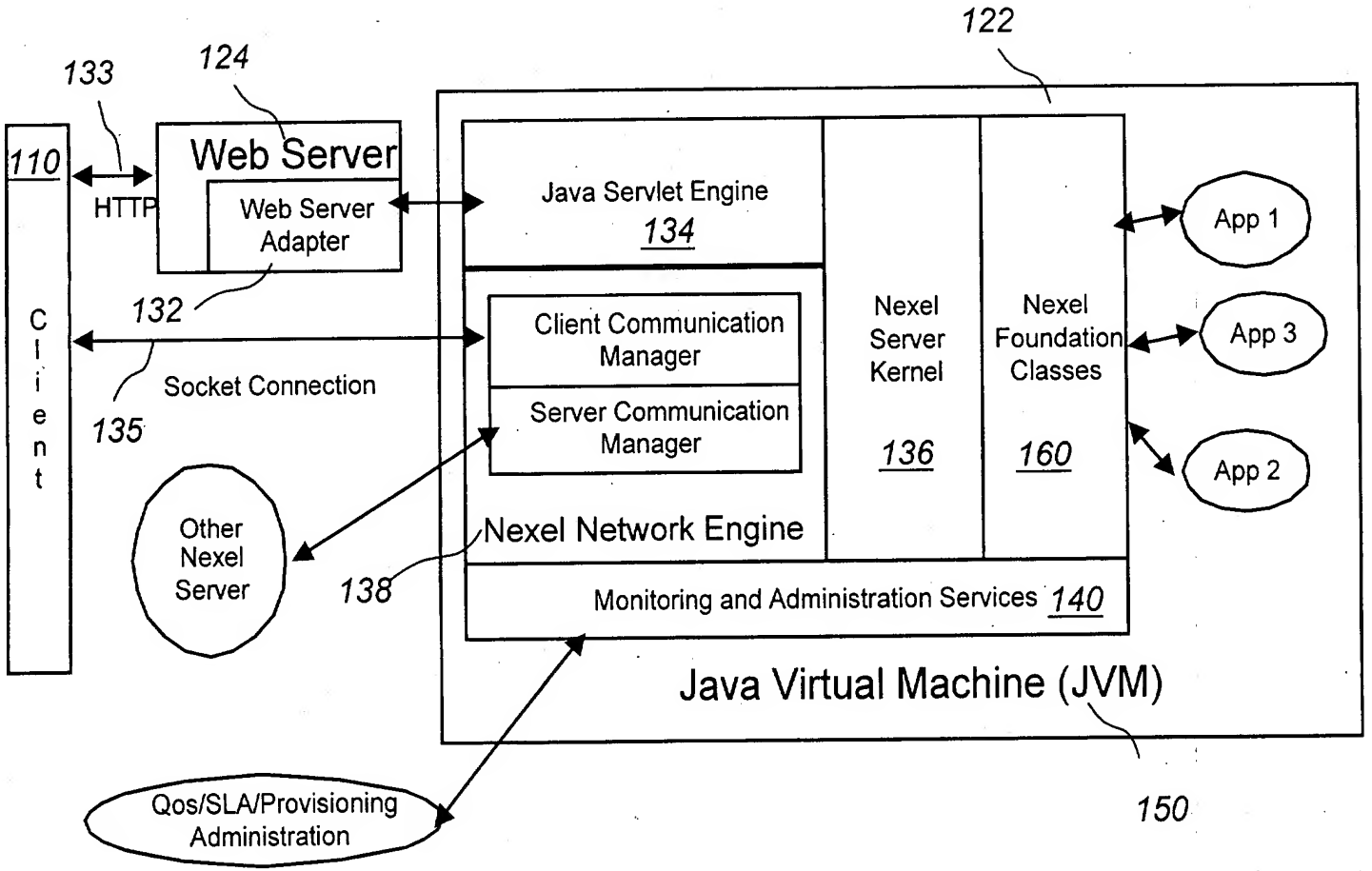
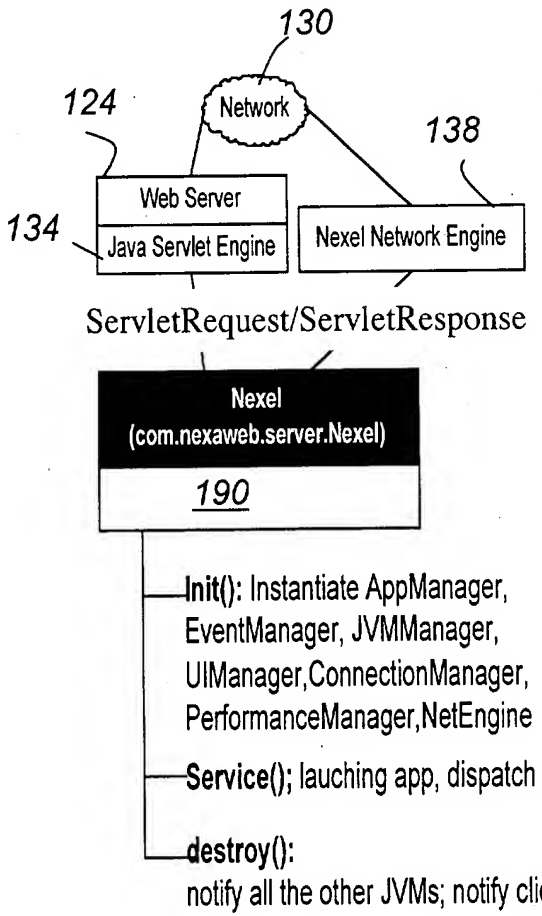


FIG. 2

PATENT & TRADEMARK OFFICE
FEB 19 2003
OIPE
JC100




```
Nexel.Service(ServletRequest, ServletResponse):
store SevletResponse to ConnectionManager;
inspect request:
        if(JVM!=thisJVM)
                pass request to NetEngine;
        else {
          if(appname!=null) {
            if(PerformaneManager.isOverLoaded())
                pass request to NetEngine;
            else launchApp();
            }
          else dispatchEvent().
          }
remove SevletResponse from ConnectionManager;
```

FIG. 3

PATENT & TRADEMARK OFFICE
FEB 19 2003
OIPE
JC100

09/354,726 - 07/08/99

10017183.021903

OIPE
FEB 1 9 2003
PATENT & TRADEMARK OFFICE

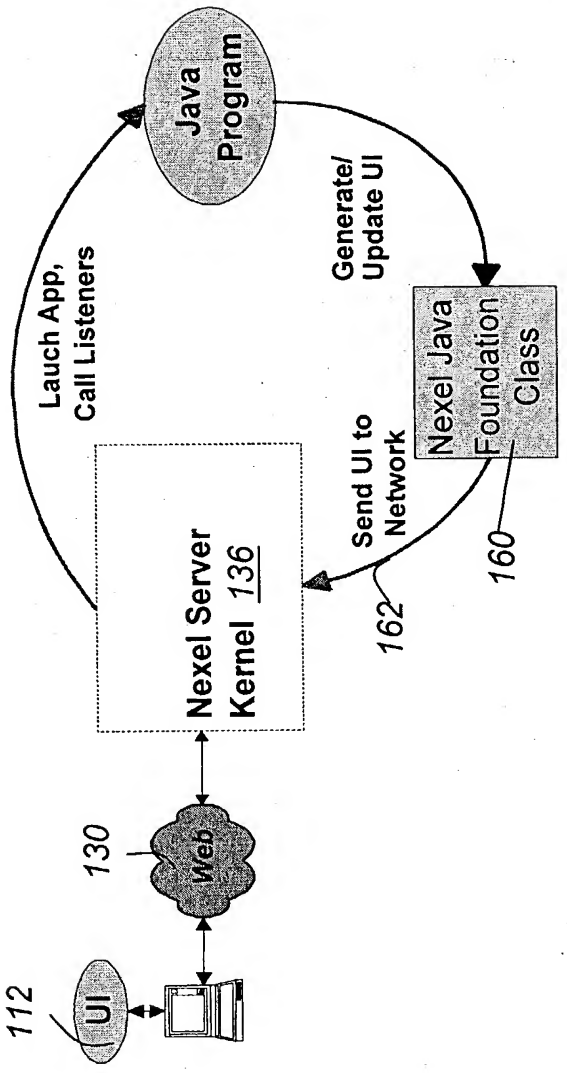


FIG. 4

OIPE JC100
FEB 1 9 2003
PATENT & TRADEMARK OFFICE

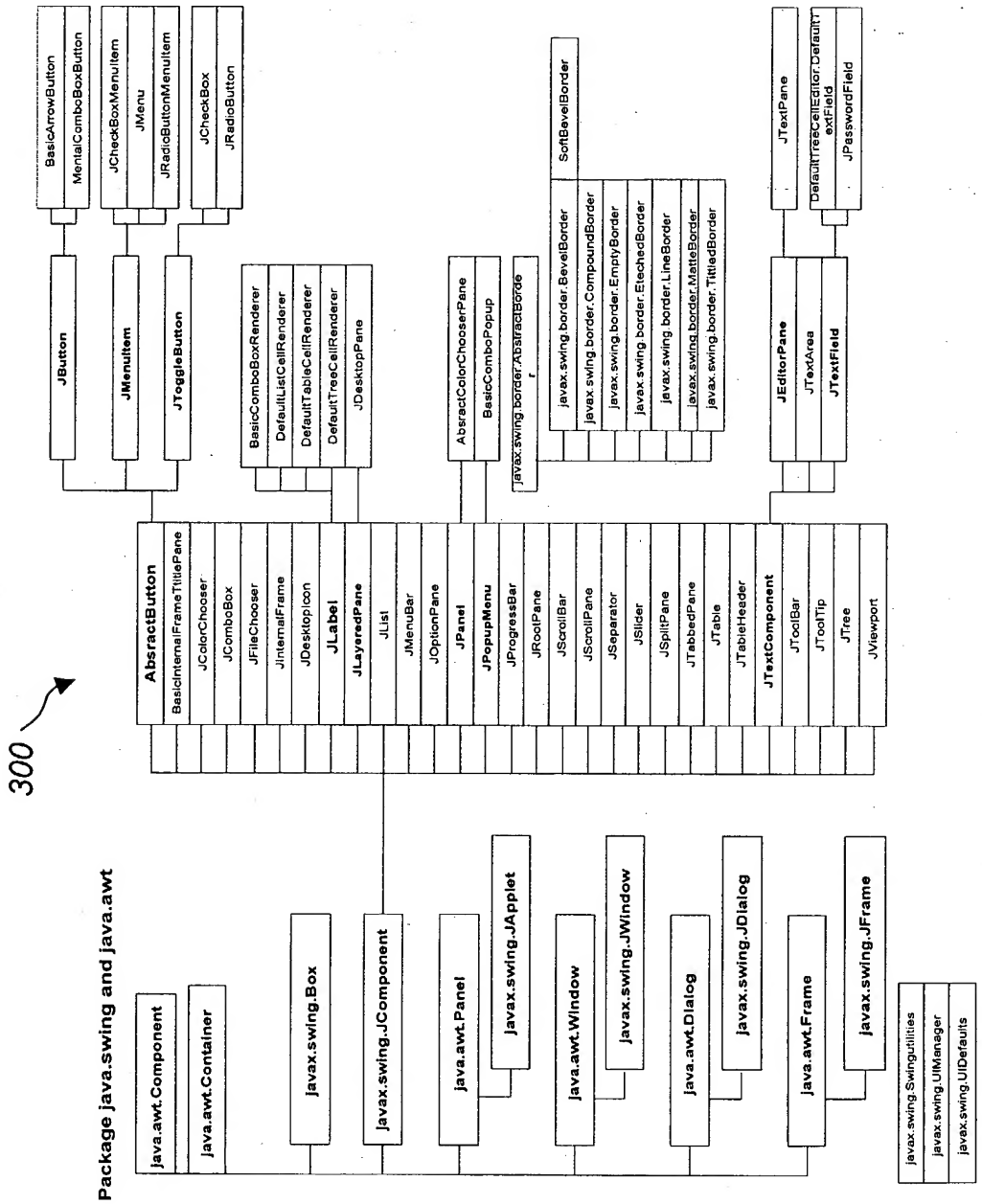


FIG. 5

10017183.021903

OIPE JC100 FEB 19 2003 PATENT & TRADEMARK OFFICE

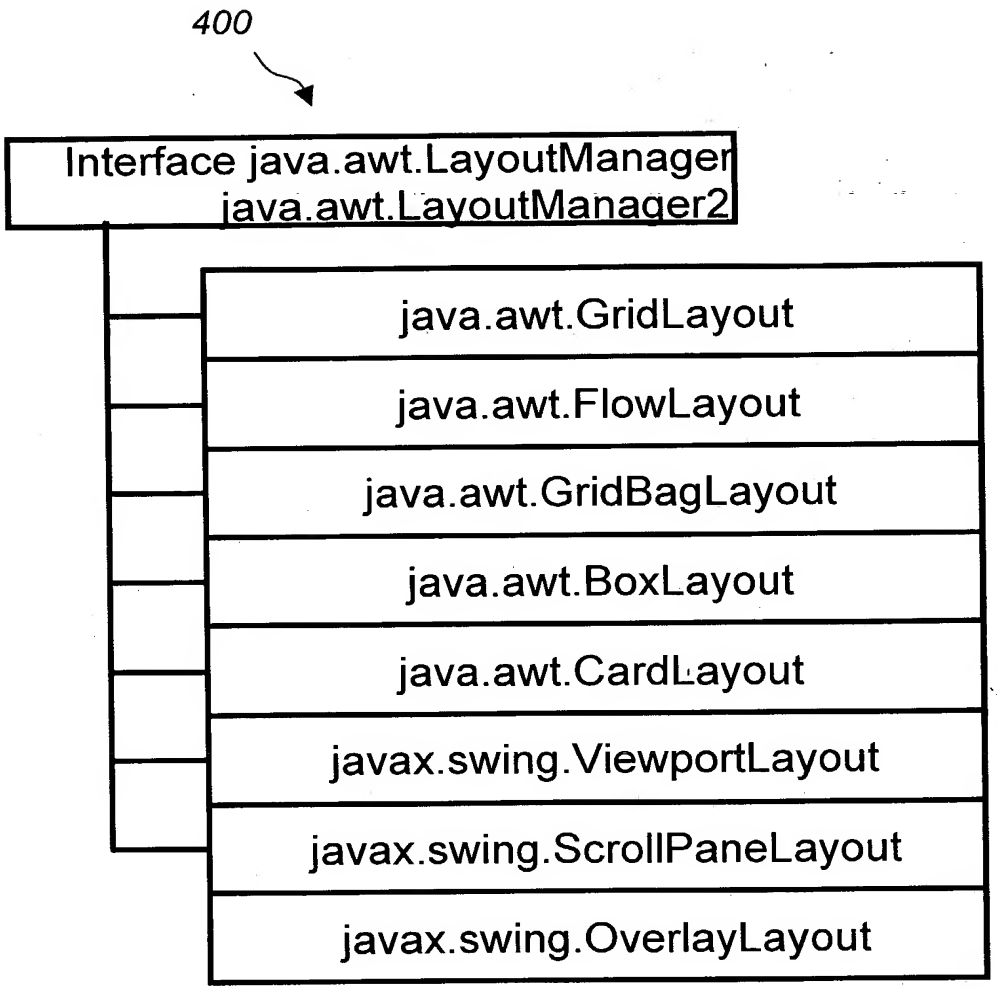


FIG. 6

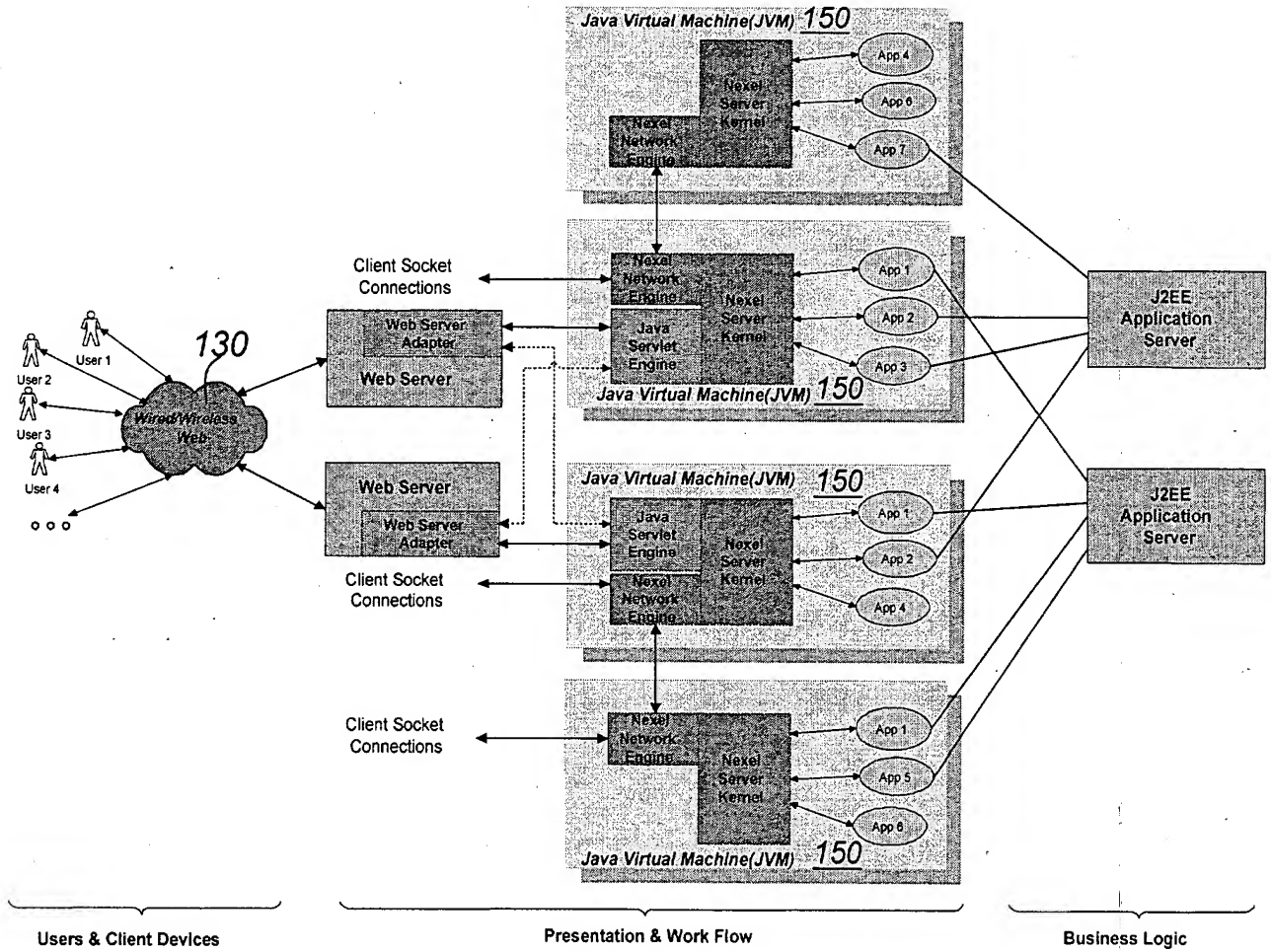


FIG. 7

PATENT & TRADEMARK OFFICE
FEB 19 2003
OIPE
JC100

09/067,529 - 02/19/03

10017183.021903

OIPE JC100 FEB 19 2003 PATENT & TRADEMARK OFFICE

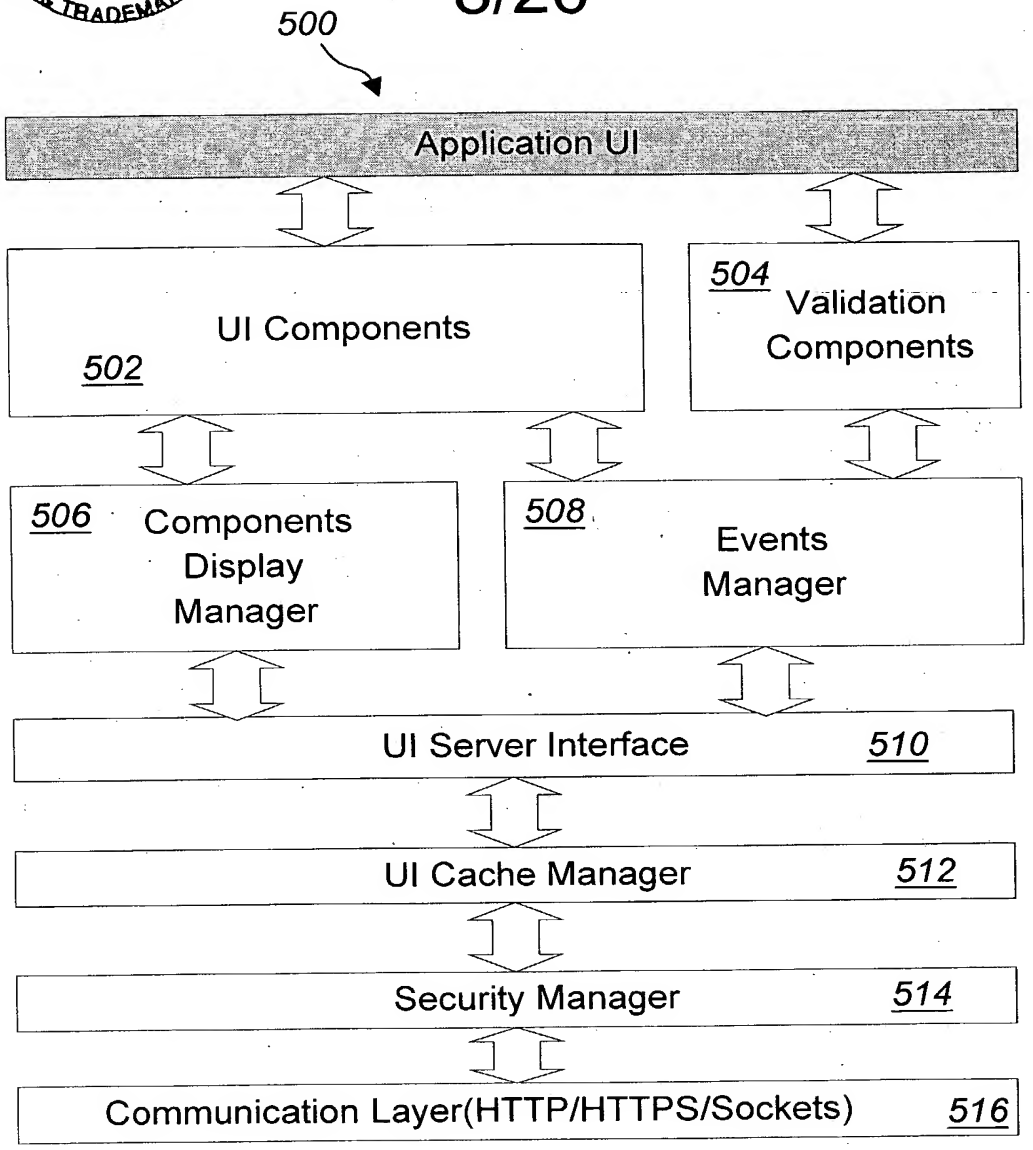


FIG. 8

10017183.021903

FEB 19 2003
PATENT & TRADEMARK OFFICE
JC100

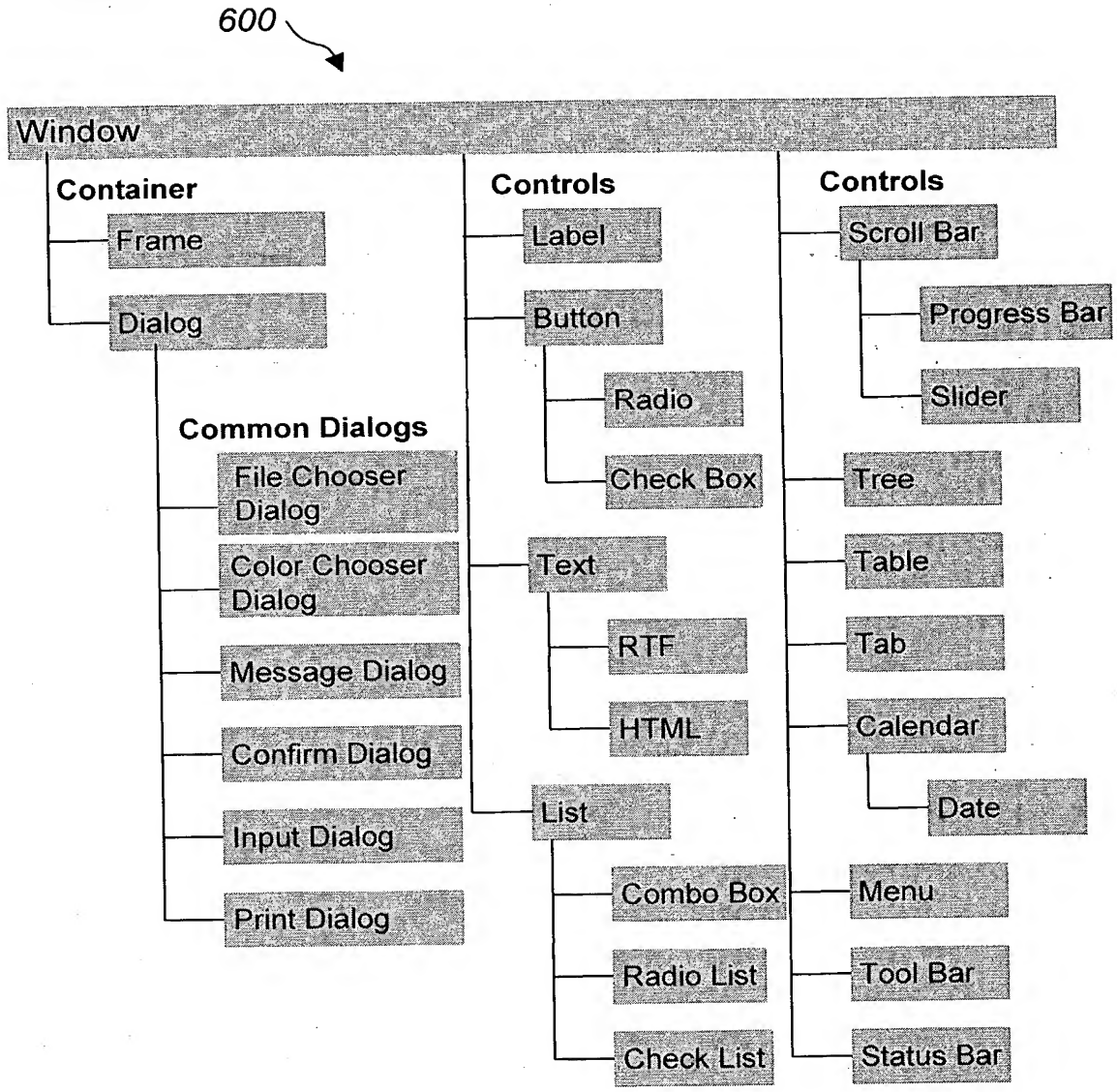


FIG. 9

OIPE JC100
FEB 1 9 2003
PATENT & TRADEMARK OFFICE

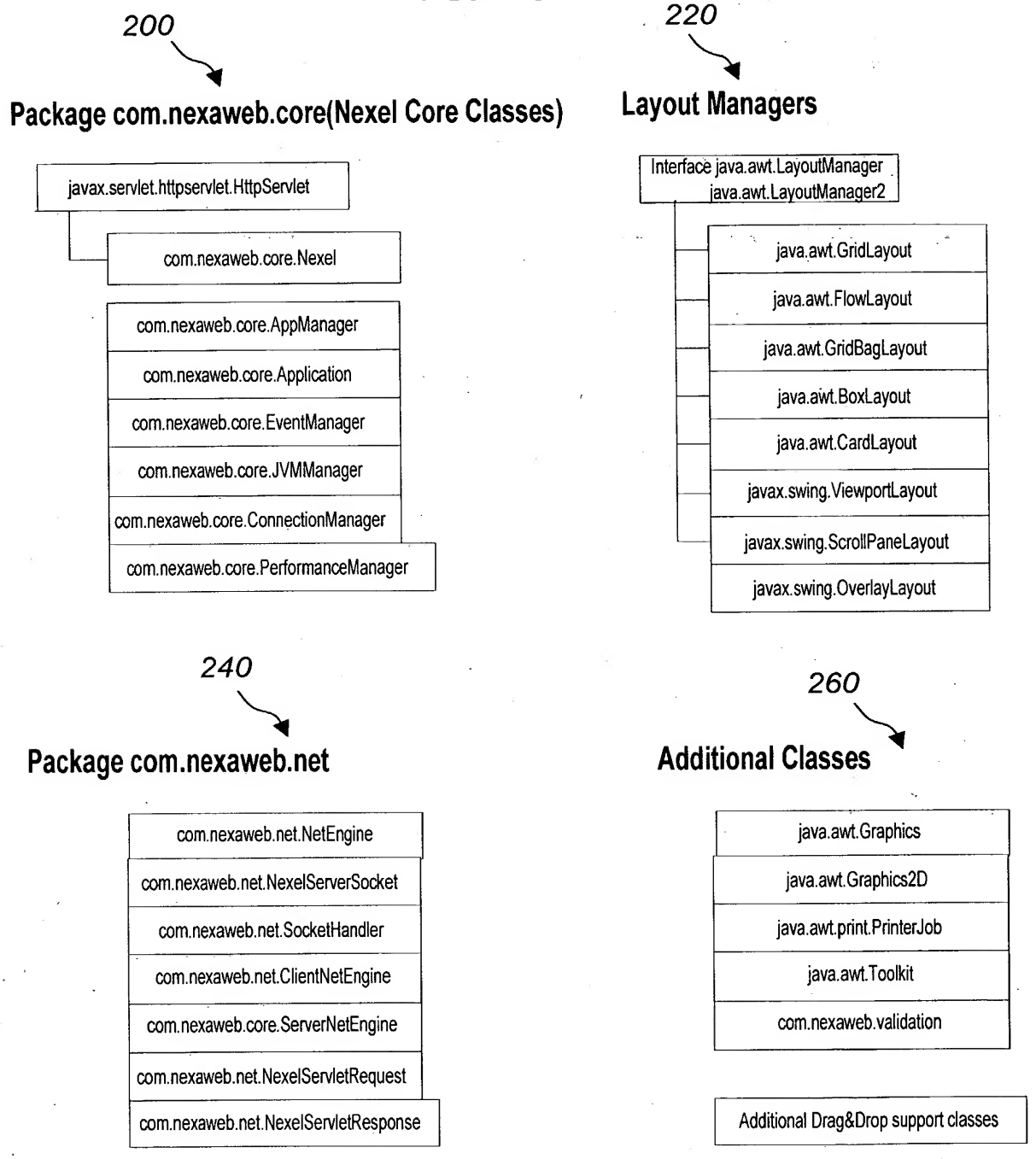


FIG. 10

10017183 .021903

OIPE JC100 FEB 19 2003 PATENT & TRADEMARK OFFICE

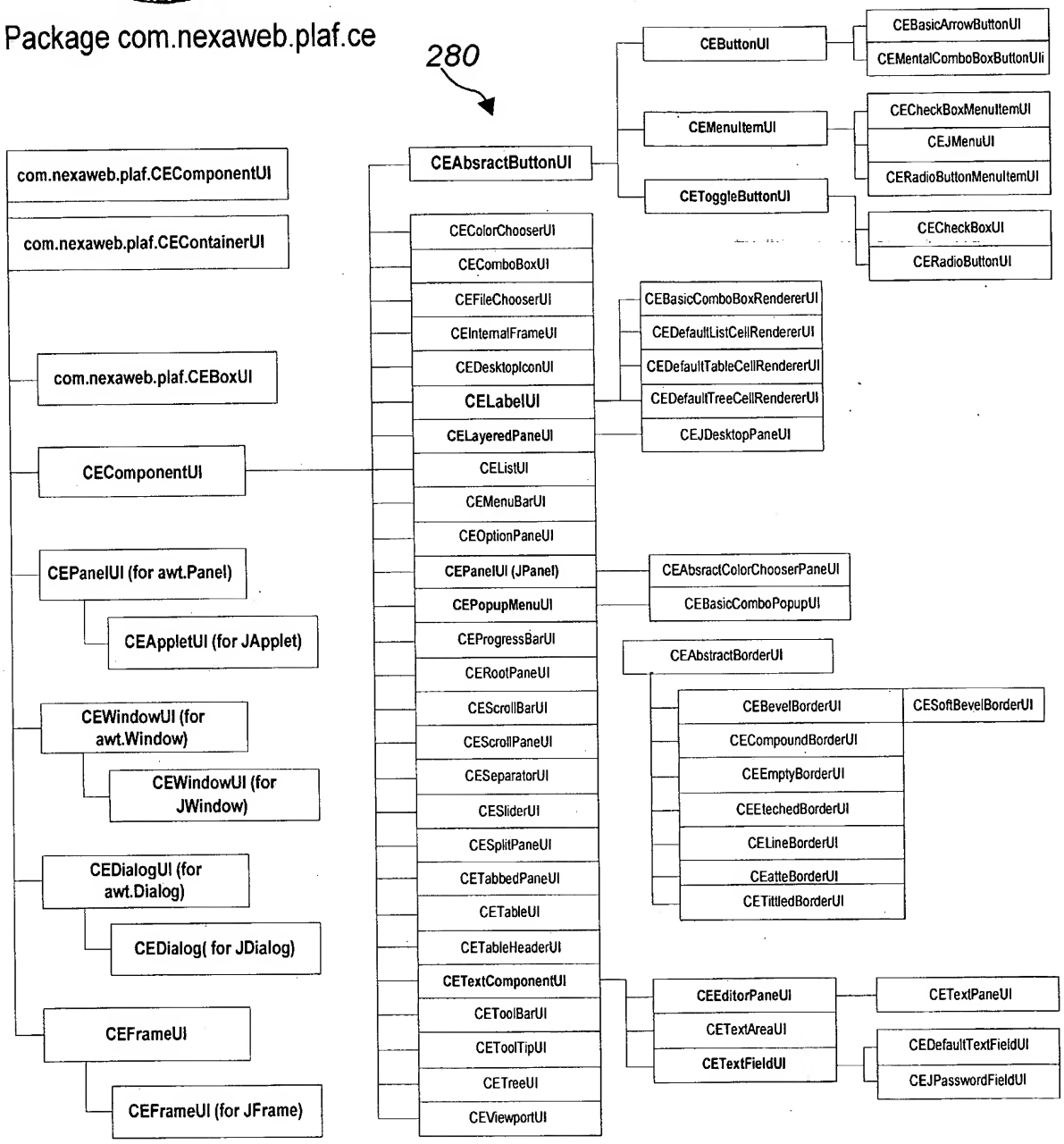


FIG. 11

10017183.021903

OTPE JC100
FEB 1 9 2003
PATENT & TRADEMARK OFFICE

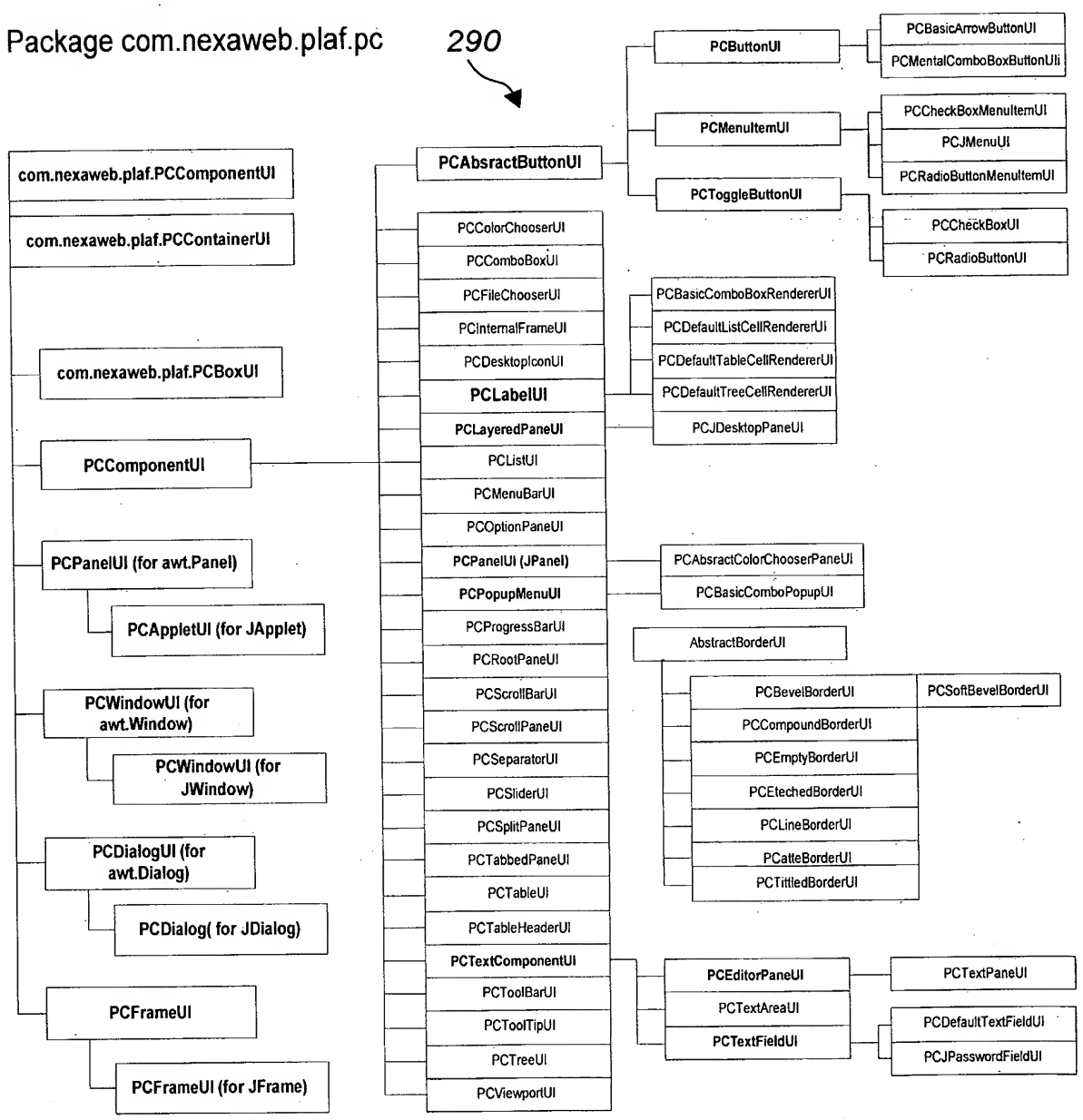


FIG. 12

OIPE JC100 FEB 19 2003 PATENT & TRADEMARK OFFICE

710

Class com.nexaweb.server.ConnectionManager

```
package com.nexaweb.server;

import java.lang.*;
import java.lang.reflect.*;
import java.util.*;
import java.io.*;
import java.text.*;
import java.awt.event.*;
import java.awt.*;

import javax.servlet.*;
import javax.servlet.http.*;

public class HttpManager {

protected static Hashtable threadList=new Hashtable();

 public  HttpManager() {return;}


public synchronized static void put(String tname, ServletResponse httpResponse)
{
  threadList.put(tname,httpResponse);
}

 public synchronized static void remove(String tname)
 {
   System.out.println("Removing entry for "+tname);
   threadList.remove(tname);
}

 public synchronized static void remove()
 {Thread th=Thread.currentThread();
  String name=th.getName();
  threadList.remove(name);
}

 private static Object getConnection(String tname) {
   System.out.println("Get connection:threadList="+threadList+",name="+tname);

   Object o=threadList.get(tname);
   System.out.println("Get connection:threadList="+threadList+",connection="+o);
   return o;
}

public static Object getConnection() {
 Thread th=Thread.currentThread();
 String name=null;
 if(th instanceof AppServiceThread) {
   /**
    *in this case, the connection is stored into HttpManager in a parent thread
    *, and the retrieving happens in a child thread
    */
     AppServiceThread ath=(AppServiceThread)th;
     name=ath.getParentThreadName();
     System.out.println("this is an AppServiceThread: parentName="+name);
}
else name=th.getName();
System.out.println("Tring to get connection by thread name="+name);
Object oo=getConnection(name);
/*Object po=oo;
   for(int i=0;true;) {break;
        if(po==null) break;
        System.out.println("Class is: "+po.getClass().getName()+"\n");
         po=po.getClass().getSuperclass();
     }
  */

 try {
 System.out.println("HttpManager:get connection:
="+oo+",oo.class="+oo.getClass().getName());
 }catch(Exception ee) {System.out.println("Exception in HTTPManager:"+ee);}
 return oo;
}

 }
```

FIG. 13

10017193 .021903

OTPE JC100 FEB 19 2003 PATENT & TRADEMARK OFFICE

720

Class com.nexaweb.server.Nexel

```
import java.io.*;
import java.text.*;
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
import com.nexaweb.server.*;


/**
 *Nexel Application Presentation Server
 *via Java Servlet Interface
 */
public class Nexel extends HttpServlet {

   public void doGet(HttpServletRequest request,
                     HttpServletResponse response)
     throws IOException, ServletException
   {

      PrintWriter out = response.getWriter();

     //App Launching Format: http://hostname:port/
servletname?appName=app1&user=user1;

      String appName=request.getParameter("appName");
      String user=request.getParameter("user");

     //App Messaging Format: http://hostname:port/
servletname?appid=appid&ctrlid=cid&key=key&eventid=eid&evparam=param
      String eid=request.getParameter("eventid");
      String appid=request.getParameter("appid");
      String cid=request.getParameter("ctrlid");

      System.out.println("Servlet
Path="+request.getServletPath()+",servername="+request.getRemoteAddr()+"
port="+request.getServerPort()+",pathInfo="+request.getPathInfo()+",URI="+re
quest.getRequestURI()+",path translated="+request.getPathTranslated());
      System.out.println("Request="+request.toString());
      System.out.println("Do post/Get:
eventid="+eid+",appid="+appid+",ctrlid="+cid);
      response.setContentType("text/html");

      out.println("<html>");
      out.println("<body bgcolor=\"lightblue\">");
      out.println("<head>");

      out.println("<title> Nexel Application Deliver Platform </title>");
      out.println("</head>");
      out.println("<body>");
      //out.println(rb.getString("requestparams.params-in-req") + "<br>");

     if(eid!=null&&eid.length()>1&&appid!=null&&appid.length()>0) {
        dispatchEvent(request,response,appid,cid,eid);
        return;
     }

      if (appName != null && user != null) {

         out.println("Application="+appName);
         out.println("user="+user);
         launchApp(request,response,appName,user);

      } else {
         //out.println(rb.getString("requestparams.no-params"));
      }
```

FIG. 14

10017183 .021903

OIPE JC100 FEB 19 2003 PATENT & TRADEMARK OFFICE

730

Class com.nexaweb.server.Nexel

```
    out.println("<P><h1>Nexel Application Delivery Platform Demo</h1>");
    out.print("<form action=\"");
    String action="http://
"+request.getServerName()+":"+request.getServerPort()+request.getRequestU
RI();

    out.print(action+"\" ");
    out.println("method=POST>");
    out.println("AppName");
    out.println("<input type=text size=20 name=appName>");
    out.println("<br>");
    out.println("User");
    out.println("<input type=text size=20 name=user>");
    out.println("<br>");
    out.println("eventid");
    out.println("<input type=text size=20 name=eventid>");
    out.println("<br>");
    out.println("appid");
    out.println("<input type=text size=20 name=appid>");
    out.println("<br>");
    out.println("Control");
    out.println("<input type=text size=20 name=ctrlid>");
    out.println("<br>");
    out.println("<input type=submit>");
    out.println("</form>");

    out.println("</body>");
    out.println("</html>");
  }

public void doPost(HttpServletRequest request,
            HttpServletResponse response)
    throws IOException, ServletException
  {
    doGet(request, response);
  }

  protected void dispatchEvent(HttpServletRequest request,
            HttpServletResponse response,String appid,String cid,String eid)
    throws IOException, ServletException
  {
    System.out.println("Dispatching event: appid="+appid+",eventid="+eid);

com.nexaweb.server.EventManager.dispatchEvent(request,response,appid,cid,
eid);
    System.out.println("Finished Dispatching event:
appid="+appid+",eventid="+eid);
  }
```

```
  protected void launchApp(HttpServletRequest request,
            HttpServletResponse response, String appName,
            String userName)
  {
    System.out.println("Launching application : "+appName);

    Thread thread=java.lang.Thread.currentThread();
    String tname=thread.getName();
    System.out.println("Working....Curent thead name ="+tname);

    Vector argsV=new Vector();
    for(int i=0;i<100;i++) //maximum arguments is 100
    {String argi=request.getParameter("apparg"+i);
    System.out.println("arguments="+argi);
     if(argi!=null) argsV.addElement(argi);
     else break;
    }

    String[] args=new String[argsV.size()];
    argsV.copyInto((Object[])args);
    Application app=new com.nexaweb.server.Application(appName,args);


    app.setBaseURL("http://
"+request.getServerName()+":"+request.getServerPort()+request.getRequestU
RI());
    System.out.println("Application Base URL="+app.getBaseURL());

    HttpManager.put(tname,response);
    System.out.println("HTTP
response="+response+",class="+response.getClass().getName());
    AppManager.addAppThread(tname,app.getAppId());


System.out.println("**********class="+response.getClass().getName()+",
of reponse?"+(response instanceof ServletResponse));
    try {
      app.start();
      System.out.println("Started Application.....");
    }catch(Exception ee) {
      System.out.println("Nexaweb Application start exception: "+ee);
    }
    HttpManager.remove(tname); //remove it after done.
    AppManager.removeAppThread(tname);

    //SimpleTest.main();
  }
}
```

FIG. 15

10017183.021903

OTPE JC100 FEB 19 2003 PATENT & TRADEMARK OFFICE

740

Class com.nexaweb.server.AppManager

```
package com.nexaweb.server;

import java.lang.*;
import java.lang.reflect.*;
import java.util.*;
import java.io.*;
import java.text.*;
import java.awt.event.*;
import java.awt.*;

import javax.servlet.*;
import javax.servlet.http.*;

/**
 * class to hold all application instances
 */
public class AppManager extends java.lang.Object {

 protected static int appCount=0;
 protected static Hashtable appTable=new Hashtable();
 protected static Hashtable appThreads=new Hashtable();

 public AppManager() {return;}

 public static String createNewAppId() {
    appCount++;
    return "Nx"+appCount+System.currentTimeMillis();
    }


 public synchronized static void addApp(Application app)
 { String key=app.getAppId();
   appTable.put(key, app);
 }


 public synchronized static void addAppThread(String tname,String appid) {
  appThreads.put(tname,appid);
 }

 public synchronized static void removeAppThread( String tname) {
  appThreads.remove(tname);
 }

 public synchronized static void removeApp(String appid)
 { Object app=appTable.get(appid); app=null;
   appTable.remove(appid);
  //appThreads.remove(appid);
 }

 public  static Application getApplication(String appid)
 { if(appid==null) return null;
   Object app=appTable.get(appid);
   return (Application)app;
 }

 public static Application getApplication()
 {
   //System.out.println("Get application");
   Thread th=Thread.currentThread();
   String tname=th.getName();
   //System.out.println("Thread name="+tname);
   String appid=(String)appThreads.get(tname);
   //System.out.println("Application ID="+appid);
   return getApplication(appid);
 }

 /**
  *a helper method to create a unique component ID for each compoent
  *(The uniqueness is only within the scope of the application)
  */
 public static String getUniqueComponentID() {
   Application app=getApplication();
   if(app==null) return "Can not find application";
   return app.getUniqueComponentID();
 }
}
```

FIG. 16

10017183.021903

OIPE JC100 FEB 19 2003 PATENT & TRADEMARK OFFICE

750

Class com.nexaweb.server.Application

```
package com.nexaweb.server;

import java.lang.*;
import java.lang.reflect.*;
import java.util.*;
import java.io.*;
import java.awt.*;


/**
 * Class to hold application information
 * This is necessary since we change the threading model of java programs. We don't maintain
 * a main thread for each application any more. Our model is a service-based model, each service
 * is served in its own thread. Once the service finished, the thread will die.
 * In order to keep different piece of an application together, we create an Application class
 * to achieve that, since different piece of the application will be handled in different threads.
 *
 */
public class Application extends java.lang.Object {
 protected String appName;
 protected String[] arguments;
 protected String appId;
 protected int componentCount=0;
 protected Hashtable listenerTable=new Hashtable();
 protected ThreadGroup group;
 protected String baseURL="";

 /**
  *A table to hold all the GUI components that belong to this application
  */
 protected Hashtable ctrlTable=new Hashtable();

 /**
  *A table to hold all other non-GUI components. This is needed when  some information
  * needs to be maintained during the entire application process, though the thread
  * that created such information may have died.
  *
  *Eacha application instance is associated with one thread group. All threads that belong
  * to this application belong to this thread group. This thread group has the same name as
  * appid.
  *
  */
 protected Hashtable dataTable=new Hashtable();
```

```
 public Application(String name, String[] args)
   {String  tname=Thread.currentThread().getName();
    appId=AppManager.createNewAppId();//tname+System.currentTimeMillis();
    //group=new java.lang.ThreadGroup(appId);
    group=Thread.currentThread().getThreadGroup();
    System.out.println("The thread group name for application "+name+"="+group.getName());
    appName=name;
    arguments=args;

    AppManager.addApp(this);
   }

 public String getAppId() {
      return appId;
 }

 public  String getUniqueComponentID() {
    componentCount++;
    return "ctrl"+componentCount;
 }

 public void setAppId(String id) {appId=id;}

 public int getNumberOfComponents() {return componentCount;}

 public ThreadGroup getThreadGroup() {
    return group;
   }

 public String getThreadGroupName() {
    return group.getName();
 }

 public void setBaseURL(String s) {baseURL=s;}
 public String getBaseURL() {return baseURL;}
```

FIG. 17

OTPE JC100 FEB 19 2003 PATENT & TRADEMARK OFFICE

760

Class com.nexaweb.server.Application

```
public synchronized void setApplicationVariable(String id, Object ctrl)
{

  if(ctrl instanceof java.awt.Component) {//need to add an ID field for
Component class
   // System.out.println("Putting CTRL="+id+", Object="+ctrl+" into the CTRL
table");
      ctrlTable.put(id,ctrl);
  }
}

public Object getApplicationVariable(String id)
{ return ctrlTable.get(id); }

public void delApplicationVariable(String id)
{ ctrlTable.remove(id); }

private  String getListenerKey(String ctrlid, String eid) {
  return ctrlid+eid;
}

public Vector getListeners(String ctrlid, String eventid) {
  String key=this.getListenerKey(ctrlid,eventid);
  Vector ls=(Vector)(listenerTable.get(key));
  return ls;
}

/**
*add a listener to be stored as application variable
*@param ctrlid, the id of the source compoent of the event
*@param eventid, the id of the event type
*@param listener, the listener object, who contains methods for event
processing
*/
public synchronized void addListener(String ctrlid, String eventid, Object
listener) {
  String key=this.getListenerKey(ctrlid,eventid);
  Vector ls=(Vector)(listenerTable.get(key));
  if(ls==null) ls=new Vector();
  ls.addElement(listener);
  listenerTable.put(key,ls);
  //System.out.println("Add Listener to Application:
ctrlid="+ctrlid+",eventid="+eventid+",listener="+listener);
}
```

```
public synchronized void removeListener(String ctrlid, String eventid, Object
listener) {
  String key=this.getListenerKey(ctrlid,eventid);
  Vector ls=(Vector)(listenerTable.get(key));
  if(ls==null) return;
  ls.removeElement(listener);
  listenerTable.put(key,ls);
}

public void start() throws Exception
  {ClassLoader cl=this.getClass().getClassLoader();
   System.out.println("entry="+appName+",appid="+appid+",class
Loader="+cl);
   Class entry;
   if(cl!=null) entry=cl.loadClass(appName);
   else entry=Class.forName(appName);

   System.out.println("entry="+appName+","+entry);

   try {
     AppServiceThread thread=new AppServiceThread(this,entry,"main",null);
   System.out.println("thread="+thread);
   thread.run();
   }catch(Exception ex) {System.out.println("Application Start Exception:
"+ex);return;}

   /**
    * thread.start();
    *We can not use thread.start() here because if you spawn off a new thread
to do the processing,
    * the original servlet service thread will just return and die. As a result, it wi
close the
    * HttpResponse connection.
    */
  }

}
```

FIG. 18

10017183.021903

OIPE JC100 FEB 19 2003 PATENT & TRADEMARK OFFICE

770

Class com.nexaweb.server.EventManager

```
package com.nexaweb.server;

import java.lang.*;
import java.lang.reflect.*;
import java.util.*;
import java.io.*;
import java.text.*;
import java.awt.event.*;
import java.awt.*;
import javax.swing.*;

import javax.servlet.*;
import javax.servlet.http.*;

public class EventManager {

 public  EventManager() {return;}

public static int getEventID(String event) {
  if(event==null) return 0;
  /**
   *mouse events
   */
  if(event.equals("MouseDown")) return 10;
  else if(event.equals("MouseUp")) return 11;
  else if(event.equals("MouseOut")) return 12;
  else if(event.equals("MouseOver")) return 13;
  else if(event.equals("MouseDoubleClick")) return 14;
  else if(event.equals("MouseClick")) return 15;
  else if(event.equals("MouseDrag")) return 16;
  else if(event.equals("MouseDrop")) return 17;
  else if(event.equals("MouseMove")) return 18;
  /**
   *Action events
   */
  else if(event.equals("ActionEvent")) return 20;
  else if(event.equals("WindowEvent")) return 30;


  else return 20000;
  }

public static int stringToInt(String s) {
     Integer eint=(new Integer(s));
     int i=0;
     if(eint!=null) i=eint.intValue();
     return i;
}

public static void dispatchEvent(HttpServletRequest request,
                HttpServletResponse response,
                String appId,String ctrlId, String eId)
     throws IOException, ServletException
  {
     System.out.println("Entering dispatch event method...");

     Application app=AppManager.getApplication(appId);
     if(app==null) {
        System.out.println("Can not find application with ID="+appId);
     }

     Vector v=app.getListeners(ctrlId, eId);
     if(v==null||v.size()<1) return;

     Thread thread=java.lang.Thread.currentThread();
     String tname=thread.getName();
     System.out.println("Working....Curent thead name
="+tname+",eventID="+eId+",ctrlID="+ctrlId);
     HttpManager.put(tname,response);
     AppManager.addAppThread(tname, app.getAppId());

     Integer eint=(new Integer(eId));
     int eventId=0;
     if(eint!=null) eventId=eint.intValue();
     System.out.println("Event ID="+eventId);

     if(eventId==getEventID("MouseDown")) {
        processMouseEvent(app, eventId,ctrlId,v,request,response);
     }
     else if(eventId==getEventID("MouseOver")) {
        processMouseEvent(app, eventId,ctrlId,v,request,response);
     }
     else if(eventId==getEventID("MouseOut")) {
        processMouseEvent(app, eventId,ctrlId,v,request,response);
     }
     else if(eventId==getEventID("MouseDown")) {
        processMouseEvent(app, eventId,ctrlId,v,request,response);
     }
     else if(eventId==getEventID("MouseUp")) {
        processMouseEvent(app, eventId,ctrlId,v,request,response);
     }
     else if(eventId==getEventID("MouseDoubleClick")) {
        processMouseEvent(app, eventId,ctrlId,v,request,response);
     }
     if(eventId==getEventID("MouseClick")) {
        processMouseEvent(app, eventId,ctrlId,v,request,response);
     }
     else if(eventId==getEventID("MouseDrag")) {
        processMouseEvent(app, eventId,ctrlId,v,request,response);
     }
     else if(eventId==getEventID("MouseDrop")) {
        processMouseEvent(app, eventId,ctrlId,v,request,response);
     }
     else if(eventId==getEventID("MouseDrag")) {
        processMouseEvent(app, eventId,ctrlId,v,request,response);
     }
     else if(eventId==getEventID("MouseMove")) {
        processMouseMotionEvent(app, eventId,ctrlId,v,request,response);
     }
     else if(eventId==getEventID("ActionEvent")) {
        System.out.println("Action Event: Calling processActionEvent");
        processActionEvent(app, eventId,ctrlId,v,request,response);
        System.out.println("Action Event: Finished processActionEvent");
     }
     else if(eventId==getEventID("WindowEvent")) {
        processWindowEvent(app, eventId,ctrlId,v,request,response);
     }

     HttpManager.remove(tname);
     AppManager.removeAppThread(tname);
     System.out.println("finished processing event. Thread="+tname);
```

FIG. 19

780

Class com.nexaweb.server.EventManager

```
    public static void  processMouseMotionEvent(Application app, int eid,String
cid,Vector listeners,HttpServletRequest request,
                HttpServletResponse response) {
                }

    public static void  processActionEvent(Application app, int eid,String
cid,Vector listeners,HttpServletRequest request,
                HttpServletResponse response) {

        System.out.println("Entering processing Action Event:
app="+app+",eid="+eid+",ctrl="+cid+",listeners="+listeners);
        if(listeners==null) return;
        if(app==null) return;

        for(Enumeration eu=listeners.elements();eu.hasMoreElements();)
         { Object o=eu.nextElement();
          if(eid==getEventID("ActionEvent")) {
                if(!(o instanceof ActionListener)) {
                    System.out.println("Listener is not of type 'ActionListener'
"+eid+",cid="+cid+",listener="+o);
                    return;
                }
                ActionListener al=(ActionListener)o;
                String cmd=request.getParameter("command");

               // System.out.println("processing action event:
command="+cmd);

                Object ctrl=app.getApplicationVariable(cid);

                //System.out.println("processing action event: CTRL="+ctrl);

                if(ctrl!=null) {
                   if(ctrl instanceof AbstractButton)
                      cmd=((AbstractButton)ctrl).getActionCommand();
                   else if(ctrl instanceof Button)
                      cmd=((Button)ctrl).getActionCommand();
                }

                String modifier=request.getParameter("modifier");
                int mask=0;
                if(modifier!=null) mask=stringToInt(modifier);
               // System.out.println("processing action event:
command="+cmd+",ctrl="+ctrl+",mask="+mask);
                ActionEvent event=new ActionEvent(ctrl,eid,cmd,mask);
                al.actionPerformed(event);
                System.out.println("processed action event...");
          }
        }
            }
```

```
    public static void  processWindowEvent(Application app, int eid,String
cid,Vector listeners,HttpServletRequest request,
                HttpServletResponse response) {
        if(listeners==null) return;
        if(app==null) return;

        for(Enumeration eu=listeners.elements();eu.hasMoreElements();)
         { Object o=eu.nextElement();
          if(eid==getEventID("WindowEvent")) {
                if(!(o instanceof WindowListener)) {
                    System.out.println("Listener is not of type 'WindowListener
"+eid+",cid="+cid+",listener="+o);
                    return;
                }
                WindowListener wl=(WindowListener)o;
                String modifier=request.getParameter("windoweventtype");
                int type=stringToInt(modifier);

                Object wo=app.getApplicationVariable(cid);
                if(!(wo instanceof Window)) {
                    System.out.println("Event source object is not of type
'java.awt.Window' "+eid+",cid="+cid+",listener="+o);
                    return;
                }
                Window win=(Window)wo;
                WindowEvent event=new WindowEvent(win,type);
                if(type==WindowEvent.WINDOW_ACTIVATED)
wl.windowActivated(event);
                else if(type==WindowEvent.WINDOW_CLOSED)
wl.windowClosed(event);
                else if(type==WindowEvent.WINDOW_CLOSING)
wl.windowClosing(event);
                else if(type==WindowEvent.WINDOW_DEACTIVATED)
wl.windowDeactivated(event);
                else if(type==WindowEvent.WINDOW_DEICONIFIED)
wl.windowDeiconified(event);
                else if(type==WindowEvent.WINDOW_ICONIFIED)
wl.windowIconified(event);
                else if(type==WindowEvent.WINDOW_OPENED)
wl.windowOpened(event);
                else {System.out.println("Window event type="+type+", is not
handled");}

                System.out.println("processed window event...");
          }
        }
            }

}
```

FIG. 20

PATENT & TRADEMARK OFFICE OIPE FEB 19 2003 JC100

09/751,896 - 122905